社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008−HPC−115 (5)
2008／5／20

ヘテロ並列計算環境における性能指標

須　田　礼　仁[†,††]

GPGPU などのアクセラレータによる高性能計算が注目を集めているが，このような計算環境はヘテロな並列計算環境とみなすことができる．しかしヘテロな並列計算環境での性能評価手法は，現在に至っても確立しているとは言い難い．本稿ではヘテロな並列計算環境の性能評価の基礎として，速度向上率と並列化効率について考察する．既存の研究における提案を概観し，それらの問題点を指摘し，解決案を提示する．

Performance Metrics for Heterogeneous Parallel Computing Environments

Reiji SUDA[†,††]

High performance computing with accelerators such as GPGPU is intensely researched. Such a system can be considered as a heterogeneous parallel processor. However, methodologies of performance evaluation for heterogeneous parallel processing systems are not established yet. This paper considers the speedup ratio and the efficiency, two basic concepts of parallel performance evaluation, for heterogeneous platforms. Reviewing existing definitions, we discuss shortcomings of those propositions, and propose a solution.

1. はじめに

近年，アクセラレータを用いた高性能計算の研究が発展を見せている．特に GPGPU の研究の広がりが目覚ましいが，歴代 GRAPE や ClearSpeed, PhysX, FPGA など，多様なものが研究されている．これらのアクセラレータは，汎用を目指す CPU とは異なり，当該ハードウェアに適した問題・アルゴリズムに対しては格段に高い性能を発揮するが，当該ハードウェアに適していない問題・アルゴリズムについては性能がほとんど出ないという特徴がある．そのため，これらのハードウェアだけで応用プログラムのすべての処理が行われるのではなく，多くの場合は汎用 CPU から制御されるコプロセッサのようになっている．性能が問題やアルゴリズムに依存するという点では，かつてスーパーコンピュータの主流であったベクトルプロセッサに似てもいるが，(1) 主記憶との結合が弱く，独自の（多くの場合小さな）ワークメモリを持っている，(2) 汎用 CPU とは異なるプログラム単位が割り当てられ，同期・通信しつつも，独自の命令制御系で CPU とは独立して動作する，という点ではベクトル演算器とは異なる特徴を有している．これらの特徴を考慮すると，CPU とアクセラレータからなるヘテロな並列計算システムとみなすのが適切である．これらの視点でみれば，CELL なども同じ範疇に入れてよい．

均一な並列計算システムに関しては，基礎概念やプログラミングの方法論は確立していると言ってよい．クラスタは広く普及し，MPI による並列プログラミングは今や常識である．効率的な並列化のために必要なのは，逐次性の回避，負荷の均衡化，通信量・回数の削減，通信遅延の隠蔽であるが，これらの知見も一般に広まり，並列化技術の基盤はすでに固まっている．これらの技術の多くは，均一なプロセッサ仕様，占有的な利用と安定した動作，適度に効率的で十分な並列性を持つ通信メカニズムなどを仮定している．これらの仮定は，効率的な並列処理を許容可能な複雑さのプログラミングで実現するために必要なものである．

しかし，ヘテロな並列計算環境に対しては既存の知見は必ずしも十分とは言えない．ヘテロな環境のための並列化手法についても研究は行われてきたが，均一な環境に比べてプログラミングの負荷が重く，高い性能を達成することも難しいために，労多くして実り少なしという感が否めない．当然，広く一般に受け入れられるということもなく，一部の変わり者の研究者が趣味でやっているという程度，と言ったら叱られるであろうか．何よりまず，高い性能が出ないというのがよくない．

ところがアクセラレータになると俄然状況が違う．場合によっては，CPU の数十倍の性能が出るという．それゆえ，とにかく注目度が高い．

ところがよく見てみると，比較対象とする CPU 側のプログラムがマルチコアに対応していないなど，性能評価に問題があることも多い．CPU と GPU の仕事の割り振りが適切かどうか，などという議論もなかなか聞かれない．やってみたらこんなに性能が出た，という報告に留まるものも少なくないようだ．

性能評価は HPC の基礎である．このままではいけな

† 東京大学 情報理工学系研究科 コンピュータ科学専攻
Dpt. of Computer Science, Grad. Schl. of Information Science and Technology, the University of Tokyo
†† CREST, JST

い．まずは，アクセラレータプロセッシングにおける性能に関する基本的な概念を整理し，性能評価の方法論を確立しなければならない．性能モデル，性能予測，性能自動チューニングへの展開，あるいは高性能プログラミングの手法や支援，さらにはアクセラレータプロセッシングを支えるコンパイラ，ミドルウェア，ライブラリ，システムソフトウェアのあり方を探る上でも，いかにして性能を評価するかは根本的な問題である．

本稿では，ヘテロな並列計算環境における並列化手法の性能評価指標についていくつか議論をする．均一な並列計算環境では，「所要時間」「高速化率」「並列化効率」などが性能評価における基本的な概念であった．ヘテロになっても「所要時間」に関しては特に新たな問題は発生しないが，「高速化率」「並列化効率」については問題が生じる．本稿では，これまでに提案されてきたヘテロ版「高速化率」「並列化効率」について概観し，それらの問題点について考察する．そして，これらの問題点を解決する評価方法の提案を行う．

2. ヘテロな並列計算環境のモデル

まず，本稿において共通に用いる表記を導入する．ある一定のまとまりの処理を表す**タスク** (**ジョブ**とも呼ばれる) は n 個あり，$T_1, T_2, \cdots, T_n$ で表す．タスク並列の場合にはタスクはスケジューリングの単位を表し，データ並列の場合にはタスクはマッピングの単位となるデータの集合と，それに付随する処理を表すものとする．また**マシン**（**プロセッサ**とも呼ばれる）は m 台あり，$M_1, M_2, \cdots, M_m$ で表す．タスク T_i をマシン M_p で処理したときの**所要時間**を t_{ip} とする．以下では n と m はそれぞれ常にタスクとマシンの数を表す．さらに，i, j はタスクのインデクス，p, q はマシンのインデクスとして使うものとする．また，「タスク T_i」とあるいは「i 番目のタスク」というべきところを，省略して「タスク i」ということにする．

2.1 計算時間のモデル

Graham ら[5] は 1979 年当時のスケジューリング理論のサーベイにおいて，並列処理における所要時間モデルを 3 つに分類している．これはスケジューリングの分野では 30 年近い歴史を有する確立した概念なので，本稿でもこれに従うことにする．

所要時間 t_{ip} がマシン p によらないとき，これらのマシンは **identical** であるという．これはホモな並列計算環境に対応する．

任意の i, p に対して所要時間が

$$t_{ip} = w_i / s_p$$

と書けるとき，これらのマシンは **uniform** であるという．w_i はタスク i の**仕事量** (work)，s_p はマシン p の**速度** (speed) と呼ばれる．

所要時間に上記のような制約を課さない場合，これらのマシンは **unrelated** であるという．したがって，ヘテロな並列計算環境は uniform な場合と unrelated の場合の 2 つに細分化されることになる．アクセラレータプロセッシングにおいては，CPU とアクセラレータの性能特性は非常に大きく異なっている．任意の処理に対して，CPU 上での所要時間とアクセラレータ上での所要時間の比が定数になるなどと期待する人はいないであろう．よってアクセラレータプロセッシングにおける性能評価では，unrelated のモデルで議論をしなければならない．Unrelated モデルは，マシン p でタスク i が処理できない $t_{ip} = \infty$ を含む．また，異なるマシンにおける所要時間の比に何の限界も設定されないので，uniform のときに定義されたタスクの「仕事量」とかマシンの「速度」とかいう概念は，unrelated という仮定では**まったく意味をなさない**ことに注意が必要である．

ヘテロ並列処理に関する従来研究のうち，特にデータ並列に関する多くのものは暗黙のうちに uniform な環境を想定している．これは理論を非常に簡単にし，アルゴリズムの構築も unrelated の場合に比べて格段に容易であるが，現実の計算機に対するあてはまりはあまりよくない．プロセッサによって異なるのは，CPU の速度だけではない．メモリアクセスのレイテンシ，キャッシュサイズ，通信性能や，複数のリクエストが来た時にこれらのリソースの実効性能がどのように影響されるかなど，要因は非常に多様である．均一なプロセッサの場合には，均一にデータを分散してやれば，メモリアクセスも，キャッシュアクセスも，通信も，すべて均一になり，各プロセッサでの所要時間は均一になってくれた．それゆえ，データを均一に分割するという手法が有効であった．しかし，不均一なプロセッサに対しては不均一なデータ分散が必要である．すると，メモリアクセスの局所性も，通信の量や回数も不均一になる．CPU の処理性能に比例する計算量を割り当てる，などという安直な方法では，ヘテロな並列計算環境の性能を十分に引き出すことはできない．

ただし，uniform という仮定が適切と思われる状況がひとつだけある．これは（マシンではなく）タスクの処理内容がすべて同一の場合である．この場合，各タスクの処理時間の逆数をマシン性能と定義できる．この場合タスクの仕事量は 1 と仮定して一般性を失わないので，このような問題を **UET** (Unit Execution Time) と呼んでいる．行列積や FFT などは均一で独立な多数のタスクに分割できるから，UET あるいは uniform というモデルがある程度有効である．

実際には厳密に uniform ではないとしても，タスクには何らかの意味で仕事量のようなものがあり，マシンにも何らかの意味で速度のようなものがあるような気がする．Unrelated という仮定は，現実からみるとあまりにも強く一般化されすぎているように感じられる．一部の研究では，所要時間を

$$t_{ip} = \sum_{k=1}^{l} w_{ik} / s_{kp}$$

のようにモデル化している．ここではタスクは l 種類の**演算** (operation) の組み合わせからなり，w_{ik} はタスク i に含まれる演算 k の仕事量，s_{kp} はマシン p が単位量の演算 k を処理する際の速度である．これはそれなりにもっともらしい仮定であり，また現実の所要時間をそれなりにモデル化できそうに思われる．このモデルでは所要時間 t_{ip} を行列とみなしたときに，この行列の階数が l になる．そこでこのモデルを本稿では **rank-l uniform** と呼ぶことにする．旧来の uniform は rank-1 uniform ということになる．このモデルは uniform と unrelated を結ぶモデルとして興味深いが，残念ながら過去の研究では一旦このように所要時間をモデル化した研究でも unrelated として扱っている．しかし最近では，GPU と CPU の性能比を論ずるのに，演算性能比とメモリアクセスバンド幅比の 2 つの要因から分析している例があり，これは rank-2 uniform モデルに相当する．このようなモデルの今後の展開に期待したい．

3. 性能指標

本稿の中心であるこの節では，ヘテロ並列計算環境における性能指標について考察する．

まず identical な場合について復習しておく．タスク全体を m 台で処理する場合の所要時間を $t(m)$ とすると，**高速化率 (speedup)** は

$$S(m) = t(1)/t(m)$$

で定義される．**理想高速化率 (ideal speedup)** は

$$S^{id}(m) = m$$

で，**並列化効率 (efficiency)** は

$$E(m) = S(m)/m = t(1)/(mt(m))$$

で定義され，

$$S(m) > m \text{ または } E(m) > 1$$

の場合には**スーパーリニアー (superlinear)** と呼ばれる．

3.1 要求要件

次にヘテロの場合について考察するが，性能指標はどのような性質のものが望ましいであろうか．いろんな意見がありうると思うが，著者は次のように考える．

(hm) いずれの指標も，identical な問題に適用したときに，identical な場合の定義に一致すること．

(sp) 高速化率は，所要時間が短いほど高い値になること．

(id) 理想高速化率は，(キャッシュヒットなどの効果を除けば) 原理的には越えられない「理想の高速化率」であって，しかし適当な条件のもとではそれに限りなく近づくことができること．

(e1) 並列化効率も，原理的に 1 が限界であって，適当な条件で 1 に近づくことができること．**(e2)** また，所要時間が短いほど高い値になること．

以下では既存の提案を概観しながら，これらの性質を満たしているかどうか検証する．

具体的な議論に先立ちいくつか記号を導入する．W_{tot} をタスク全体，W_p をマシン p に割り当てられたタスクの集合とし，マシン (1 台，あるいはマシンの集合) M でタスクの集合 W を処理するときの所要時間を $t(W, M)$ とする．また，M_{tot} をヘテロ並列計算環境全体，M_{ref} を高速化率を評価する際の**基準マシン (reference machine)**, M_{max} を $t(W_{tot}, M_p)$ を最小にするマシン（**最速マシン, fastest machine**）とする．また，タスク集合 W の「仕事量」を $|W|$ で表す（unrelated の場合には問題がある）．

3.2 Uniform な場合

Uniform な場合には問題はそれほど難しくはない．まず高速化率について考えると，基準になるようなマシン M_{ref} があれば

$$S_{ref} = t(M_{ref})/t(M_{tot})$$

で，そうでなければ最速のマシン M_{max} を基準にして

$$S_{max} = t(M_{max})/t(M_{tot})$$

と定義すればよい．以下 M_{ref} を使う場合と M_{max} を使う場合とでほとんど平行して議論ができるので，まとめて M_* と書くことにする．すなわち 2 つの定義は

$$S_* = t(M_*)/t(M_{tot})$$

とまとめられる．これらは条件 (hm), (sp) を満たす．

Uniform の場合にはマシンの速度 s_p が定義できる．**合計マシン速度 (total processor speed)** を

$$s_{tot} = \sum_{p=1}^{m} s_p$$

と定義すれば**等価マシン数 (equivalent number of processors)**[3] が

$$m_* = s_{tot}/s_*$$

のように定義される．これは基準マシン M_* の性能を 1 としたときの，ヘテロ並列計算環境全体の計算性能をあらわす．Identical であれば $m_* = m$ となる．

これからすると，理想速度向上率は

$$S_*^{id} = m_*$$

と定義するのがよいように思われる．これに従えば

$$S_* > m_*$$

の場合にスーパーリニアーと呼び，並列化効率は

$$E_* = S_*/m_*$$

と定義されることになる．Zhang ら[11] は M_{max} を用いたこれらの定義を使用しているが，これは条件 (hm), (id), (e1), (e2) をすべて満たす．著者は M_{ref} を用いた定義を見たことはないが，この場合には条件 (id), (e1) を満たさない．

Donaldson ら[4] は

$$S_{max} > m$$

をもってスーパーリニアーと定義している．マシンの**数**というものを重視すれば，これも可能な選択肢である．対応した並列化効率は

$E^{\#}_{max} = S_{max}/m$

で定義されることになるが，これは条件 (e1) を満たさない．他方，Mazzeo ら[7] は

$S_{ref} > m$

は**スーパーリニアーではない**と書いている（当然だ）．

関連する概念として，Bazterra ら[1] は **effective number or processors** というものを

$$m_{eff} = \sum_p t(W_p, M_p)/t(W_{tot}, M_{tot})$$

で定義している．Uniform の場合には

$$\max_p\{t(W_p, M_p)\} \le t(W_{tot}, M_{tot}) \le \sum_p t(W_p, M_p)$$

と仮定すれば $t(W_p, M_p) = |W_p|/s_p$ より

$1 \le m_{eff} \le m$

および $|W_{tot}| = \sum_p |W_p|$ より

$S_{max} \le m_{eff}$

となることがわかる．Bazterra の意図は，高速化率とともにこれも指標のひとつとして評価の対象にしてはどうかということであった．しかし m_{eff} を高速化率とみなすと，条件 (sp) を満たさない．実際，速度が 1, 1, 3 のマシンがあったとき，速度 1 のマシンにタスクを半分ずつ割り当てる方が，速度 3 のマシンにすべてのタスクを割り当てるよりも高い m_{eff} となる．

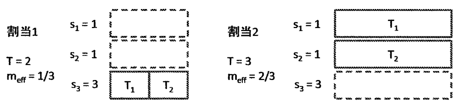

図 1 m_{eff} が (sp) を満たさない例

Effective number of processors が意味するところは，CPU 数と平均稼働率の積である．つまり

$E^T = m_{eff}/m$

は**平均稼働率**を表しているが，これは identical の場合には並列化効率に一致し，並列化効率の拡張の一種とみなすことができる．しかしこれは条件 (e1) を満たさない．

以上をまとめると，Zhang らが用いている S_{max}, S^{id}_{max}, E_{max} が条件をすべて満たすので，uniform での性能指標として適切である．

3.3 Unrelated な場合

Unrelated な環境では，タスク i をマシン p が処理できないことがある．極端な場合，タスク全体 W_{tot} をすべて実行できるマシンが 1 台も存在しないという場合すら考えられる．

仮に上記のような極端な状況でないとすると，M_{max} や M_{ref} が意味を持つ．すると高速化率が

$S_* = t(W_{tot}, M_*)/t(W_{tot}, M_{tot})$

と定義できる．このときマシン数ベースの並列化効率

$$E^{\#}_{max} = S_{max}/m$$

およびマシン数ベースのスーパーリニアー

$S_{max} > m$

も容易に定義できる[4]．

しかしこの並列化効率は条件 (e1) を満たさない．これは次の例により証明できる．仕事は独立な 2 つのタスクからなり $W_{tot} = \{T_1, T_2\}$, マシンは 2 台 $M_{tot} = \{M_1, M_2\}$ であるとする．所要時間を

$t(T_1, M_1) = 1, \qquad t(T_1, M_2) = 3$

$t(T_2, M_2) = 3, \qquad t(T_2, M_2) = 1$

とする．タスク全体を 1 台で処理させると $t(W_{tot}, M_1) = t(W_{tot}, M_2) = 4$ となり，いずれのマシンでも 4 の時間がかかる．しかし T_1 を M_1 に，T_2 を M_2 に割り当ててやれば makespan は 1 である．つまり，$S_{max} = 4$ となり，2 台で 4 倍の速度向上が得られたことになる．Mechoso らの論文[8] はこのようなスーパーリニアーが実際のアプリケーションで得られることを強調した論文である．アクセラレータプロセッシングによる性能向上も，これと本質的に同じことであると考えられる．

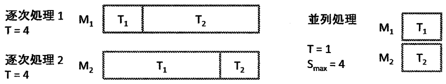

図 2 $E^{\#}_{max}$ が (e1) を満たさない例

何人かの研究者は，理想速度向上率から並列化効率とスーパーリニアを定義することを試みた．それらはまずマシンの速度を定義することからスタートしている．絶対的なマシンの速度を定義する試みとしては，Colombet ら[3] による

$$s^{loc}_p = |W_p|/t(W_p, M_p)$$

および Pastor ら[9] による

$s^{tot}_p = |W_{tot}|/t(W_{tot}, M_p)$

がある．前者は値が $|W_p|$ の定義次第でどうにでもなってしまうという問題があるが，タスク全体を処理できるマシンが 1 台もなくても定義できる．

相対的な速度を定義する試みもある．Zhang ら[11] は

$r^{totmax}_p = t(W_{tot}, M_{max})/t(W_{tot}, M_p)$

を，Clematis ら[2] は

$$r^{totref}_p = t(W_{tot}, M_{ref})/t(W_{tot}, M_p)$$

を，Mazzeo ら[7] は

$$r^{locref}_p = t(W_p, M_{ref})/t(W_p, M_p)$$

をそれぞれ定義している．

次にこれらの速度に基づいて理想速度向上率を定義した例を見る．Zhang ら[11] は

$$S^{id}_{totmax} = \sum_p r^{totmax}_p = \sum_p \frac{T(W_{tot}, M_{max})}{T(W_{tot}, M_p)}$$

を理想速度向上率と定義した．しかし先の例をこれで評価すると $S^{id}_{totmax} = 2$ となってしまう．実際の速度向上率は $S_{max} = 4$ であるから，条件 (id) を満たしていない．Clematis ら[2] は

$$S^{id}_{totref} = \sum_p r^{totref}_p$$

として，Pastor ら[9] は $\sum_p s^{tot}_p / s^{tot}_{ref}$ として理想速度向上率を定義しているが，これらでも同じ結果になる．

他方，Mazzeo ら[7] は

$$S^{id}_{locref} = \sum_p r^{locref}_p = \sum_p \frac{t(W_p, M_{ref})}{t(W_p, M_p)}$$

を提案している．先の例では $S^{id}_{locref} = 4$ となり，S_{ref} に一致して具合がよい．これは一般に成り立ち，対応する並列化効率

$$E_{locref} = \frac{S^{id}_{locref}}{S_{ref}} = \frac{t(W_{tot}, M_{ref})/t(W_{tot}, M_{tot})}{\sum_p t(W_p, M_{ref})/t(W_p, M_p)}$$

は

$$t(W_{tot}, M_{ref}) = \sum_p t(W_p, M_{ref})$$

$$t(W_{tot}, M_{tot}) \geq t(W_p, M_p) \qquad \forall p$$

という自然な仮定のもとで

$$0 \leq E_{locref} \leq 1$$

が証明できる．すなわち条件 (e1) を満たす．Colombet らの論文[3] の前半でもこれとほぼ等価な $\sum s^{loc}_p / s^{tot}_{ref}$ で理想速度向上率を定義している．

しかしこれらの定義は条件 (e2) を満たさない．先の例で所要時間モデルをわずかに変更して $t(T_2, M_2) = 2$ とする．すると所要時間は 2 となり，並列化効率は

$$E_{locref} = \frac{4/2}{1/1 + 3/2} = \frac{4}{5}$$

となる．しかし逆に T_1 を M_2 に，T_2 を M_1 に割り当てると所要時間は 3 に伸びてしまうが，並列化効率は

$$E_{locref} = \frac{4/3}{1/3 + 3/3} = 1$$

となる．この原因はマシンの性能指標が割り当てられたタスク W_p に依存して定義されているところにある．

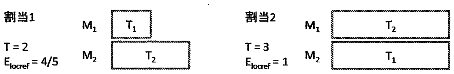

図 3　E_{locref} が (e2) を満たさない例

Effective number of processors m_{eff} は unrelated でも定義でき，並列化効率

$$E^T = m_{eff}/m$$

も定義できるが，やはり上記の例により (e2) を満たさないことが示される．

以上のような提案の根本的な問題は，unrelated であるにも関わらずマシンに「速度」などという概念を付与しようとしたところにある．これを打破するためには，速度という概念を用いずに，直接**理想所要時間** t^{id} を定義し，

$$E = t^{id}/t(W_{tot}, M_{tot})$$

により並列化効率を定義すればよい．スーパーリニアーは $E > 1$ として自然に定義できる．

このような定義が Colombet の論文[3] の後半にある（以下の説明はオリジナルの定式化とは若干異なるが，本質的に同じである）．タスクのマシンへの割り当ての単位であるタスクに立ち戻り，割り当て変数 x_{ip} の整数制約を取り除いた

$$\begin{aligned}
&\text{minimize} \quad t \\
&\text{subject to} \quad \sum_i x_{ip} t_{ip} = t \qquad \forall p \\
&\qquad\qquad\qquad \sum_p x_{ip} = 1 \qquad \forall i \\
&\qquad\qquad\qquad 0 \leq x_{ip}
\end{aligned}$$

という線形計画問題を考える．この問題の解の t は所要時間の下限になっており，identical や uniform の場合の理想所要時間に一致する．また，前述の理想速度向上率の諸定義のような問題は生じない．これは条件 (hm), (e1), (e2) をすべて満たす．

問題としては，線形計画問題を解く手間がかかるという点が挙げられる．Colombet は uniform の場合に簡単に解けることを指摘しているが，実は rank-2 uniform でも高速に解ける（紙数の都合により省略）．

以下，本節をまとめる．並列化効率には Colombet による線形計画の解に基づく定義を用いることを提唱する．高速化率 S_* は条件 (hm), (sp) を満たすが，すべてのタスクを処理できるマシンがないと定義できない点に注意が必要である．理想所要時間があれば理想高速化率は必要ないと思うが，あえて定義するなら $t(W_{tot}, M_*)/t^{id}$ とすればよい．

4. 最悪性能比：まとめにかえて

速度向上率や並列化効率は，ホモな古典的な並列計算環境でよく用いられてきた性能指標である．その背景にあるのは，マシン数というコストを計算時間という意味でいかに効率よく使用するか，というコスト・パフォーマンス比としての効率であったと思われる．

しかしながら，これらの指標の評価において通信コストは「オーバーヘッド」（悪）にしかならない．このため，環境によってはアルゴリズムをいくら工夫してもこれらの指標が理想的な値から遠いままという場合も多い．性能指標を上げなければならない，という意識が働くと，粒度が大きく独立なタスクからなる「自明な問題」を取り上げることになってしまう．これらの問題が悪いとか意味がないとかいうことではないが，このような問題ばかりを取り上げてしまうようでは今後の研究の発展に悪影響を与えるのではないかと危惧するものである．

これに対して，スケジューリング理論[5),10)] では**最悪性**

能比 (**worst case performance ratio**) をアルゴリズムの性能指標として用いることが多い．これは与えられた問題に対する最適解の makespan C_{max}^{opt} に対して，近似解の makespan C_{max}^{app} がどれだけ近いかという比の上界である．つまり任意の問題に対して

$$C_{max}^{app}/C_{max}^{opt} \leq r$$

となるような r（定数の場合も，問題サイズなどの関数の場合もある）を求め，それをアルゴリズムの性能指標として利用するのである．なお，並列化効率と比べると分母と分子が逆である点に注意してほしい．性能が高いほど小さい値になり，理想値は 1 である．

通常は最適解は未知であるため，最適解の下界を分母にして性能比の上界を求めるのが一般的である．前節で取り上げた Colombet らの線形計画問題の解は下界のひとつであるが，Lawler ら[6] によるプリエンプティブスケジューリングの解も理想所要時間を決定するのに用いることができる．これは

$$\begin{aligned} &\text{minimize} \quad t \\ &\text{subject to} \quad \sum_p \tau_{ip}/t_{ip} = 1 \qquad \forall i \\ &\qquad\qquad\quad \sum_i \tau_{ip} \leq t \qquad \forall p \\ &\qquad\qquad\quad \sum_p \tau_{ip} \leq t \qquad \forall i \\ &\qquad\qquad\quad \tau_{ip} \geq 0 \end{aligned}$$

の最適解 t で与えられる（τ_{ip} はマシン p でタスク i を処理する時間である）．先の例で所要時間を

$$t(T_1, M_1) = 3, \qquad t(T_1, M_2) = 3$$
$$t(T_2, M_1) = 2, \qquad t(T_2, M_2) = 1$$

とすると，どのようにタスク割り当てを工夫しても所要時間は 3 より短くならないから，例えば T_1 を M_1 に，T_2 を M_2 に割り当てれば最適解である．ところが前節の Columbet らによる下界は $t^{id} = 2$ となり「並列化効率」は 2/3 にしかならない．これに対し，最悪性能比で評価すれば理想値 1 となる．

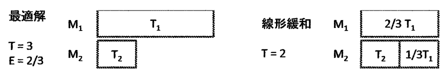

図 4　最適解でも $E = 1$ が達成できない例

このように，問題設定や並列アルゴリズムを固定した場合の並列化技術に関する評価については，この最悪性能比という指標が重要である．とりわけ，最悪性能比による評価では，通信コストが大きくて速度向上率が上がらないような問題設定であっても，最適に近い並列化を行えばよい指標が得られるという点で，今後のネットワーク計算環境上での並列化技術の評価基準としては重要であろうと考える．

実際には並列処理に向いた問題のモデル化や定式化，並列処理に向いたアルゴリズムなどの開発も極めて重要な課題であるが，これらを評価する際には最悪性能比は適していない．並列化効率と最悪性能比の両方の特性を理解し，両者を適切に使い分けて評価と研究を進めることが重要である．

謝辞

本研究の一部は文部科学省科学研究費，JST CREST，Microsoft IJARC CORE project の支援を受けています．

参 考 文 献

1) Bazterra, V. E., Cuma, M., Ferraro, M. B. and Facelli, J. C.: A general framework to understand parallel performance in heterogeneous clusters: Analysis of a new adaptive parallel genetic algorithm, *J. Par. Dist. Comp.*, Vol. 65, No. 1, pp. 48–57 (2005).
2) Clematis, A. and Corana, A.: Modeling performance of heterogeneous parallel computing systems, *Par. Comp.*, Vol. 25, No. 9, pp. 1131–1145 (1999).
3) Colombet, L. and Desbat, L.: Speedup and efficiency of large-size applications on heterogeneous networks, *Theoret. Comput. Sci.*, Vol. 196, No. 1–2, pp. 31–44 (1998).
4) Donaldson, V., Berman, F., and Paturi, R.: Program speedup in a heterogeneous computing network, *J. Par. Dist. Comp.*, Vol. 21, No. 3, pp. 316–322 (1994).
5) Graham, R. L., Lawler, E. L., Lenstra, J. K. and Rinnooy Kan, A. H. G. R.: Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals of Discrete Mathematics*, Vol. 5, pp. 287–326 (1979).
6) Lawler, E. L. and Labetoulle, J.: On preemptive scheduling of unrelated parallel processors by linear programming *J. ACM*, Vol. 25, No. 4, pp. 612–619 (1978).
7) Mazzeo, A., Mazocca, N. and Villano, U.: Efficiency measurements in heterogeneous distributed computing systems: From theory to practice, *Concurrency: Pract. Exper.*, Vol. 10, No. 4, pp. 285–313 (1998).
8) Mechoso, C. R., Farrara, J. D. and Spahr, J. A.: Achieving superlinear speedup on a heterogeneous, distributed system, *IEEE Par. Dist. Tec.: Sys. App.*, Vol. 2, No. 2, pp. 57–61 (1994).
9) Pastor, L. and Bosque, J. L.: Efficiency and scalability models for heterogeneous clusters, *3rd Int'l Conf. Cluster Comp.*, pp. 427–434 (2001).
10) 須田礼仁：ヘテロ並列計算環境のためのタスクスケジューリング手法のサーベイ，情報処理学会論文誌：コンピューティングシステム，Vol. 47, No. SIG 18 (ACS 16), pp. 92–114 (2006).
11) Zhang, X.-D. and Yan, Y.: Modeling and characterizing parallel computing performance on heterogeneous networks of workstations, *Proc. 7th IEEE Symp. Par. Dist. Proc.*, pp. 25–34 (1995).